RUMOR DETECTION FROM SOCAIL MEDIA

A Mini Project Report

submitted by

Ayisha Beeba(MES20MCA-2013)

to

the APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the Degree

of

Master of Computer Applications

Department of Computer Applications

MES College of Engineering
Kuttippuram, Malappuram - 679 582

February 2022

DECLARATION

I undersigned hereby declare that the project report **Rumor Detection From Socail Media**, submitted for partial fulfillment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala, is a bonafide work done by me under supervision of Dr Geever C Zacharias, Assistant Professor, Department of Computer Applications. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place:

....

Date:

DEPARTMENT OF COMPUTER APPLICATIONS
MES COLLEGE OF ENGINEERING, KUTTIPPURAM

CERTIFICATE

This is to certify that the report entitled **Rumor Detection From Social Media** is a bonafide record of the Mini Project work carried out by **Ayisha Beeba(MES20MCA-2013)** submitted to the APJ Abdul Kalam Technological University, in partial fulfillment of the requirements for the award of the Master of Computer Applications, under my guidance and supervision. This report in any form has not been submitted to any other University or Institution for any purpose.

Internal Supervisor(s) External Supervisor(s)

Head Of The Department

Acknowledgements

It is great pleasure to acknowledge the support of many people who have contribute to the successful completion of this project. This project would never have seen the light of support, co- operation and valuable guidance extended by Dr Geever C Zacharias. Also IWould like to sincere gratitude to Dr Geever C Zacharias for her valuable advice and kind of encouragement as internal guide during the whole process from time to time. She always been a source of inspiration.Without their encouragement and help this project would not have materialized.I'm also grateful to all our teaching and non-teaching staff for their encouragement, guidance and whole-hearted support. Last but not the least; I would like to express my sincere thanks to our Family member and our friends for their constant encouragement.

Ayisha Beeba(MES20MCA-2013)

Abstract

n this era, social media platform are increasingly used by people to follow newsworthy events because it is fast, easy to access and cheap comparatively. Despite the increasing use of social media for information and news gathering, its nature leads to the emergence and spread of rumours i.e., information that are unverified at the time of posting, which may causes serious damage to government, markets and society. Therefore, there is necessity of effective system for detecting rumours as early as possible before they widely spread. Effective system should consist of four components: Rumour detection, rumour tracking, stance classification, and veracity classification. Lots of work has been done in later component while very less work in component rumour detection. So, now we should work on rumour detection. In this paper, we will summarise efforts done till now in this area. Most of existing methods detects a priori rumours, i.e., predefined rumours. So it is required to have automated rumour detection method which detects new emerging rumours effectively and as early as possible.

Keywords:

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Nowadays, people are using social media networks to share their ideas, opinions, and feeling. Also, access to news is effortless and comfort with the using of social media networks which in the past people had to use newspapers and magazines to get aware of the world situations, but now they are using from social media networks to read the latest news just in a minute after a bad or good news occurred in the world. people are addicted to reading the news by using social media networks, which is the easiest way for them, but one issue that sometimes decreases the popularity of social media is dealing with Rumor. Here the main work is to seek the best outcome to find and detect Rumor and misleading news from social media networks with the help of deep learning by analyzing the similarity of the news

1.1.1 Motivation

Rumour detection is considered as binary classification task where we have predefined set of category of binary class as Rumour, Non-Rumour and labelled dataset is there to train classifier. Binary classification is a category of classification that classifies the events into two categories based on features. Binary classification would generally fall in the domain of supervised learning since dataset is labelled.

1.2 Objective

• Providing easiest method to find a news is Rumor or not • Using deep learning to make the output more accurate • Awareness about Rumor news on social media

1.3 Report Organization

The project report is divided into four sections. Section 2 describes literature survey. Section 3 describes the methodology used for implementing the project. Section 4 gives the results and discussions. Finally Section 5 gives the conclusion.

Chapter 2

Literature Survey

2.1 Introduction

There has been very little work done in automatic detection of new emerging rumour. Most existing method detects a priori rumour (e.g., Obama is muslim) where classifier is feed with predefined rumour, then classifier can classify post based on keyword(Obama and muslim) of predefined rumours. We study and analyse existing method to detect rumour in social media and we represent summary of all that methods in this section.

2.2 Evolution Measures

This method learns RNN models by utilizing the variation of aggregated information across different time intervals related to each event. RNN-based method can be evaluated with three widely used recurrent units, tanh, LSTM and GRU, which perform significantly better.

Chapter 3

Methodology

3.1 Introduction

Collection of data from post datasets using Twitter (www. twitter.com) For the Twitter data, we confirmed rumors and non-rumors from www. snopes.com, an online rumor debunking service. For each event, we extract the keywords from the last part of the Snopes URL, e.g., http://www.snopes.com/pentagon-spends-powerballtickets. We refine the keywords by adding, deleting or replacing words manually, and iteratively until the composed queries can have reasonably precise Twitter search results. Apply a type of feed-forward neural network RNN that can be used to model variable-length sequential information such as sentences or time series. The RNN-based model will classify posts or microblog events into rumors and non-rumors. As follows Firstly converts the incoming streams of posts as continuous variable-length time series. Then describe RNNs with different kinds of hidden units and layers for classification. This method learns RNN models by utilizing the variation of aggregated information across different time intervals related to each event. Empirically evaluate the RNN-based method with three widely used recurrent units, tanh, LSTM and GRU, which perform significantly better.

3.2 Deep Learning Method

Deep learning is the new scope of machine learning, which uses artificial neural networks to produce excellent accuracy in tasks, like speech recognition, object detection, and language translation. Moreover, the terrific strength of deep learning is that it can automatically learn and translate features from images, videos.

3.3 Modules

ADMIN • View users • View feedback • View complaint and send reply • Block /Unblock users USERS • Update profile • Add and Manage friend request • Send friend request • View post and news • Add news and post • Add and view comments • Send complaint and view reply • Send feedback

Chapter 4

Results and Discussions

4.1 Datasets

Post/Blog datasets using Twitter (www. twitter.com)

4.2 Developing Environment

4.2.1 Hardware Requirement

- Processor : Intel Pentium Core i3 and above, 64 bits
- RAM : Min 3GB RAM
- HARD DISK: 10 GB

4.2.2 Software Requirements

- OPERATING SYSTEM: WINDOWS 10
- FRONT END: HTML, CSS, JAVASCRIPT
- BACK END: Mysql
- IDE USED: Jetbrains Pycharm, Android studio
- TECHNOLOGY USED: PYTHON JAVA

- FRAME WORK USED: Flask

4.3 Future Enhancement

Using deep learning and artificial intelligence we can find out the genuieneness of a news.In this project when someone post a news in our social media the system will find out that news is rumor or not by using deep learning.So user doesn’t need to research about the news.

Chapter 5

Conclusions

Generally rumours spread hatred or fear which is extremely harmful to society. So, we must take some steps to diffuse this rumour. In this paper, we summarised psychological study of rumour, existing methods to detect rumour, and future enhancement used to evaluate performance of method. Research in rumour detection is growing day by day as use of social media is increasing in society. As existing methods are not such capable that can efficiently process stream data and automatically detect new emerging rumours from social media, so we need a complete system that can automatically detect new emerging rumours as early as possible.

5.1 References

[1][Allport and Postman, 1965] G.W. Allport and L.J. Postman.The psychology of rumor. Russell Russell, 1965. [2] [Bengio et al., 1994] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2):157–166, 1994. [3] [Castillo et al., 2011] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on twitter. In Proceedings of WWW, 2011. [4] [Cho et al., 2014] Kyunghyun Cho, Bart van Merrienboer, ¨ Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014.

References

[1] **Andrew, Ng** (2011) Sparse Autoencoder, [Online]. Available: `https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf`.

[2] **Krogh, A. and Hertz, J.A.** (1992) A simple weight decay can improve generalization, Advances in Neural Information Processing Systems, 950-957.

[3] **Gonzalez, R.C. and Wintz, P.** Digital Image Processing, 2nd Edition, Addison-Wesley, 1987.

[4] **Bazen, A.M. and Gerez, S.H.** (2001) Segmentation of fingerprint images, *ProRISC 2001 Workshop on Circuts, Systems and Signal Processing*, Veldhoven, The Netherlands.

Appendix

Source Code

```
 # This piece of software is bound by The MIT License (MIT)
# Copyright (c) 2013 Siddharth Agrawal
# Code written by : Siddharth Agrawal
# Email ID:siddharth.950@gmail.com

from flask import *

from src.db_connection import *

app=Flask(__name__)
app.secret_key="qwetrt"


import functools
def login_required(func):
   @functools.wraps(func)
   def secure_function():
      if "lid" not in session:
         return redirect ("/")
      return func()
   return secure_function





@app.route('/')
def start():
   return render_template('ADMIN_login.html')

@app.route('/login',methods=['post'])
def login():
   username=request.form['textfield']
   password=request.form['textfield2']
   query="select * from login where Username=%s and Password=%s"
   value=(username,password)
   res=selectonenew(query,value)
   if res is None:

      return '''<script>alert(" invalid data");window.location="/"</script>'''
   elif res[3]=="admin":
      session['lid']=res[0]
      return '''<script>alert(" login success");window.location="/adminhome"</script>'''
   else:
      return '''<script>alert(" invalid");window.location="/"</script>'''
```

Appendix

```
@app.route('/adminhome')
def adminhome():
   return render_template('ADMIN_home.html')
@app.route('/adminviewuser')
@login_required
def adminviewuser():

      query="SELECT `user_registration`.*,`login`.* FROM `login`JOIN `user_registration` ON
            `user_registration`.`lid`=`login`.`Lid`"
      res=selectall(query)

      return render_template('ADMIN_VIEWuser.html',val=res)


@app.route('/adminviewnews')
@login_required
def adminviewnews():

      qry="SELECT
            newspost.*,user_registration.fname,user_registration.mname,user_registration.lname,user_registration.emailid
            FROM newspost JOIN user_registration ON user_registration.lid=newspost.uid"
      res=selectall(qry)
      print(res)
      return render_template('ADMIN_VIEWnews.html',val=res)
      # return "okkk"

@app.route('/deletenews',methods=['get'])
def deletenews():
   id = request.args.get('id')
   qry="DELETE FROM `newspost` WHERE `nid`=%s"
   value=(str(id))
   iud(qry,value)
   return '''<script>alert("deleted");window.location="/adminviewnews"</script>'''

@app.route('/adminviewfeedback')
@login_required
def adminviewfeedback():

      qry="SELECT `feedback`.*,`user_registration`.`fname`,`mname`,`lname` FROM `feedback` JOIN `user_registration` ON
            `feedback`.`uid`=`user_registration`.`lid`"
      res=selectall(qry)
      return render_template('ADMIN_VIEWfeedback.html',val=res)

@app.route('/adminviewchatbot')
def adminviewchatbot():
   qry="SELECT * FROM `dataset`"
   res=selectall(qry)
   return render_template('ADMIN_VIEWchatBOT.html',val=res)
@app.route('/adminaddnewchatbot',methods=['post','get'])
def adminaddnewchatbot():
   return render_template('ADMIN_ADDnewCHATBOT.html')
@app.route('/adminaddnewchatbotnew',methods=['post'])
def adminaddnewchatbotnew():

   QUESTION=request.form['textfield']
   ANSWER=request.form['textfield2']
   qry="INSERT INTO`dataset` VALUES(NULL,%s,%s)"
   val=(QUESTION,ANSWER)
   iud(qry,val)
   return '''<script>alert("success");window.location="/adminaddnewchatbot"</script>'''
@app.route('/deleteqstn',methods=['get'])
def deleteqstn():
   id=request.args.get('id')


   qry="DELETE FROM `dataset` WHERE `cid`=%s"
   value=(str(id))
   iud(qry,value)
   return '''<script>alert("deleted");window.location="/adminviewchatbot"</script>'''




@app.route('/blockuser',methods=['get'])
def blockuser():
   id = request.args.get('id')

   qry="UPDATE `login` SET TYPE='blocked' WHERE `Lid`=%s"
   value=(str(id))
   iud(qry,value)

   return '''<script>alert("blocked");window.location="/adminviewuser"</script>'''
@app.route('/unblockuser',methods=['get'])
def unblockuser():
   id = request.args.get('id')

   qry="UPDATE `login` SET TYPE='user' WHERE `Lid`=%s"
   value=(str(id))
   iud(qry,value)

   return '''<script>alert("unblocked");window.location="/adminviewuser"</script>'''


@app.route('/view_complaint_and_send_reply')
@login_required
def view_complaint_and_send_reply():
   qry="SELECT `complaint`.*,`user_registration`.`fname`,`user_registration`.`lname` FROM `user_registration` JOIN
         `complaint` ON `complaint`.`userid`=`user_registration`.`lid` WHERE `complaint`.`reply`='pending'"
   s=selectall(qry)
   return render_template('view_complaint_annd_send_reply.html',val=s)
@app.route('/send_reply')
def send_reply():
   id=request.args.get('id')
   session['lid']=id
   return render_template("send_reply.html")

@app.route('/reply_send',methods=['post'])
def reply_send():
   cid=session['lid']
   reply=request.form['textarea']
   qry="UPDATE `complaint` SET `reply`=%s WHERE `cid`=%s"
   val=(reply,str(cid))
   iud(qry,val)
   return '''<script>alert("success");window.location="/view_complaint_and_send_reply"</script>'''

@app.route('/logout')
def logout():
   session.clear()
   return render_template("ADMIN_login.html")

app.run(debug=True)
```

5.2 Source Code

import os

import nltk import numpy as np from flask import* from nltk import word$_{t}okenizefromnltk.corpusim$
$fromsrc.chatbotimportcb$

app=Flask($_{name}{}_{)@app.route('/login',methods=['post'])deflogin():print(request.form)importrequests}$

username=request.form[’uname’] password=request.form[’pass’] qry=”select*from ‘login‘ where Username=val=(username,password) s=selectonenew(qry,val)

if s is None: return jsonify(’task’:’invalid’) else: id=s[0] return jsonify(’task’:’valid’,”id” : id)

@app.route(’/reg’,methods=[’post’]) def reg(): print(request.form) fname=request.form[’FN’] mname=request.form[’MN’] lname = request.form[’LN’] phno = request.form[’PN’] emailid = request.form[’EI’] gender = request.form[’GN’] photo = request.files[’files’] file=secure$_{f}ilename(photo.$
$request.form['DOB']place = request.form['PLACE']district = request.form['DS']$

username = request.form[’UN’] password = request.form[’PS’] confirmpassword=request.form[’CPS’] if password==confirmpassword: qry=”INSERT INTO‘login‘ VALUES(NULL,value=(username,password) s=iud(qry,value) qry=”INSERT INTO ‘user$_{r}egistration`VALUES(NULL, val = (fname, mname, lna$
$returnjsonify('tsak' :' failed')@app.route('/postnews', methods = ['post'])defpostnews() :$
$print(request.form)uid = request.form['uid']heading = request.form['heading']content =$
$request.form['content']iflen(content) > 15 : print(content)res = checknews(content)print("result$
$"real" : content = content.replace("'", " : ", ".", ",", " - ", "`")$

content = content.replace(”’”, ” ”) content = content.replace(”:”, ” ”) content = content.replace(”.”, ” ”) content = content.replace(”,”, ” ”) content = content.replace(”-”, ” ”) content = content.replace(”‘”, ” ”) content = content.replace(”;”, ” ”)

qry=”INSERT INTO‘newspost‘ VALUES(NULL,value = (uid,heading,content) iud(qry, value) return jsonify(’task’: ’success’) else: return jsonify(’task’: ’error’) else: return jsonify(’task’: ’invalid’)

@app.route(’/viewnews’,methods=[’post’]) def viewnews(): qry=”SELECT * FROM newspost” res=androidselectallnew(qry) return jsonify(res) @app.route(’/sharenews’,methods=[’post’]) def sharenews(): print(request.form) uid = request.form[’uid’] fromid = request.form[’fromid’] toid = request.form[’toid’] userid = request.form[’userid’] qry=”INSERT INTO‘sharenews‘

VALUES(NULL,value = (uid, fromid,toid,userid) iud(qry, value) return jsonify('task': 'success')

@app.route('/addfeedback',methods=['post']) def addfeedback(): print(request.form) uid = request.form['uid'] feedback = request.form['feedback'] qry="INSERT INTO‘feedback‘VALUES(NULL
iud(qry,value) return jsonify('task': 'success') @app.route('/viewfeedback',methods=['post']) def viewfeedback(): uid=request.form['uid'] print(uid) qry="SELECT ‘user$_r$egistration‘.∗, ‘feedback‘.∗
FROM‘feedback‘JOIN‘user$_r$egistration‘ON‘user$_r$egistration‘.‘lid‘ = ‘feedback‘.‘uid‘WHERE‘f
value = (uid)res = androidselectall(qry, value)print(res)returnjsonify(res)

@app.route('/viewmynews',methods=['post']) def viewmynews(): uid=request.form['uid'] qry="SELECT * FROM‘newspost‘ WHERE uid=value=(uid) res = androidselectall(qry, value) print(res) return jsonify(res)

@app.route('/viewmy$_f$riends$_n$ews', methods = ['post'])defviewmy$_f$riends$_n$ews() : uid =
request.form['uid']qry = "SELECT‘newspost‘.∗FROM‘newspost‘JOIN‘friend$_r$equest‘ON‘news
‘friend$_r$equest‘.‘FROMID‘WHERE‘friend$_r$equest‘.‘toid‘ = value = (uid, uid)res =
androidselectall(qry, value)

return jsonify(res)

@app.route('/deletemynews',methods=['post']) def deletemynews(): nid = request.form['nid'] qry="DELETE FROM‘newspost‘WHERE nid=value = (nid) iud(qry, value) return jsonify('task': 'success')

@app.route('/viewfriend$_r$equest', methods = ['post'])defviewfriend$_r$equest() : qry =
"SELECT∗FROM‘friend$_r$equest‘"res = androidselectallnew(qry)returnjsonify(res)

@app.route('/viewfriend$_r$equest', methods = ['post'])defviewfriend$_r$equest() : fromid =
request.form['fromid']print(fromid)qry = "SELECT‘friend$_r$equest‘.∗, ‘user$_r$egistration‘.‘fname
‘user$_r$egistration‘.‘lid‘WHERE‘friend$_r$equest‘.‘toid‘ = value = (fromid)res = androidselectall(q
0 : print("no")returnjsonify("no")else : print("ys")returnjsonify(res)

@app.route('/viewfriend$_r$equest$_c$ount', methods = ['post'])defviewfriend$_r$equest$_c$ount() :
fromid = request.form['fromid']print(fromid)qry = "SELECTcount(∗)FROM‘user$_r$egistration
‘user$_r$egistration‘.‘lid‘WHERE‘friend$_r$equest‘.‘toid‘ = value = (fromid)res = selectonenew(qry,

@app.route('/acceptfriend$_r$equest', methods = ['post'])defacceptfriend$_r$equest() : fid =
request.form['fid']qry = "UPDATE.‘friend$_r$equest‘SET‘status‘ =' accepted'WHERE‘fid‘ =
value = (fid)iud(qry, value)returnjsonify('task' :' success')@app.route('/rejectfriend$_r$equest', me

['post'])defrejectfriend$_r$equest() : fid = request.form['fid']qry = "UPDATE.'friend$_r$equest'SET

rejected'WHERE'fid' = value = (fid)iud(qry, value)returnjsonify('task' :' success')

@app.route('/viewprofile',methods=['post']) def viewprofile(): uid = request.form['uid'] print(uid) qry=" SELECT 'photo','fname' ,mname,lname FROM'user$_r$egistration'WHERElid = value = (uid)res = androidselectall(qry, value)print(res)returnjsonify(res)

@app.route('/send$_f$riendrequest', methods = ['post'])defsend$_f$riendrequest() : uid = request.form['uid']print(uid)qry = "SELECT'user$_r$egistration'.'fname', 'user$_r$egistration'.'lnam

value = (uid, uid, uid)res = androidselectall(qry, value)print(res)returnjsonify(res)@app.route('/

['post'])defviewmore$_f$riend() : qry = "SELECT ∗ FROM'user$_r$egistration'"res = androidselectallnew(qry)returnjsonify(res)

@app.route('/send$_f$riendrequest2', methods = ['post'])defsend$_f$riendrequest2() : fromid = request.form['fromid']toid = request.form['toid']qry = "INSERTINTO'friend$_r$equest'VALUES

(fromid, toid)iud(qry, value)returnjsonify('task' :' success')

@app.route('/viewfriends',methods=['post']) def viewfriends(): lid=request.form['uid'] print(lid) qry="SELECT 'friend$_r$equest'.∗, 'user$_r$egistration'.∗FROM'user$_r$egistration'JOIN'friend

'user$_r$egistration'.'lid'WHERE'friend$_r$equest'.'toid' = value = (lid, lid)res = androidselectall(qr

@app.route('/in$_m$essage2', methods = ['post'])defin$_m$essage() : print(request.form)fromid = request.form['fid']print("fromid", fromid)

toid = request.form['toid'] print("toid",toid)

message=request.form['msg'] print("msg",message) qry = "INSERT INTO 'chat' VALUES(NULL,value = (fromid, toid, message) print("ppppppppppppppppp") print(value) iud(qry, value) return jsonify(status='send')

@app.route('/view$_m$essage2', methods = ['post'])defview$_m$essage2() : print("wwwwwwwwwwww

request.form['fid']print(fromid)toid = request.form['toid']print(toid)lmid = request.form['lastm

lmid)sen$_r$es = []qry = "SELECT∗FROMchatWHERE(fromid = qry = "SELECT'fromid', 'm

print("SELECT'fromid', 'message', 'date', 'msgid'FROM'chat'WHERE'msgid' > val = (str(lmid), str(toid), str(fromid), str(fromid), str(toid))print("fffffffffffff", val)res = androidselectall(qry, val)print("resullllllllll")print(res)ifresisnotNone : returnjsonify(status =

ok', res1 = res)else : returnjsonify(status =' notfound')

@app.route('/userchat',methods=['post']) def userchat(): fromid = request.form['fromid'] toid = request.form['toid'] message=request.form['message'] qry = "INSERT INTO 'chat'

VALUES(NULL,value = (fromid, toid, message) iud(qry, value) return 'success'

@app.route('/viewchat',methods=['post']) def viewchat(): fromid=request.form['fromid'] toid=request.form['toid'] print(fromid,toid) qry="SELECT * FROM chat WHERE (fromid=val=(str(fromid res = androidselectall(qry,val) return jsonify(res) @app.route('/searchnews',methods=['post']) def searchnews(): print(request.form) heading = request.form['heading'] content = request.form['content'] print(content) res = checknews(content) print("resultttttttttttttttt") print(res)

qry = "INSERT INTO'newspost' VALUES(NULL, value = (uid, heading, content) iud(qry, value) return jsonify('task': res)

@app.route('/insertchatbot',methods=['post']) def insertchatbot(): qus = request.form['msg'] lid = request.form['lid'] print(lid)

res = cb(qus) qry = "INSERT INTO 'chatbot' VALUES(NULL,val=(str(lid),qus,res) iud(qry,val) return jsonify('task': "ok")

@app.route('/response',methods=['post']) def response(): st$_i d = request.form['lid']$

qry = "SELECT msg,u$_i d, answerFROM`chatbot`WHERE`u_i d` = val = (str(st_i d))s = selectallnew(qry, val)$

val=(st$_i ds = select1(qry, val)row_h eaders = ['frmid','toid','msg']json_d ata = []forresultins :$ $row = []row.append(st_i d)row.append(0)row.append(result[0])row.append(result[3])json_d ata.append$ $[]row.append(0)row.append(st_i d)row.append(result[2])row.append(result[3])json_d ata.append(dict(z$

def checknews(news):

res = stop(news) key = " i = 1 resultset = [] resultset1 = []

for r in res: key = key + " " + r keyval = key

print('https://www.google.co.in/search?rlz=1C1CHBF$_e nIN790IN790biw = 935bih = 657ei = CDVcXLOCJJeRwgPorKjwDAq =' +keyval)$

res1 = requests.get('https://www.google.co.in/search?rlz=1C1CHBF$_e nIN790IN790biw = 935bih = 657ei = CDVcXLOCJJeRwgPorKjwDAq =' +keyval)print(res1.text)resn = []$

print(res1.text) import re clean = re.compile('¡.*?¿')

print(res1.text) class="ZINbbc xpd O9g5cc uUPGi" ll = res1.text.split('¡div class="ZINbbc xpd O9g5cc uUPGi"') print(ll) print(len(ll), "lennnnnnnnnnnnnnnnnnnnnnnn") for i in range(1, len(ll)-1): ll1 = ll[i] print(type(ll1)) lll=ll1.split('¡div class="kCrYT"¿') if len(lll)¿2:

newsl=lll[2] print(newsl) print("==

text = re.sub(clean, ””, newsl) print(text) print(”==

resn.append(text) print(resn, ” urlsssssssssssssssssss”)

sim = [] for n in resn: print(n)

dictl = process(n)

print(”dictl”,n,dictl)

dict2 = process(news) print(”dict2”,news,dict2)

sim.append(getsimilarity(dict2, dictl)) print(”==”)

print(”similarity between Bug599831 and Bug800279 is ”, sim) sum = 0.0 cou = 0 print(”sim”,sim)

for s in sim: if float(s) ¿ 0.45: cou = cou + 1 sum = sum + float(s)

sum = sum / len(sim) conn = cou / len(sim) print(cou / len(sim)) print(sum) thr = ”” if conn

¿= 0.5: thr = ”real” else: thr = ”fake” cmd.execute(”insert into news values(null,’” + str(uid)

+ ”’,’” + heading + ”’,’” + news + ”’,curdate(),’”+thr+”’)”) con.commit() print(thr) return thr

def getsimilarity(dictl, dict2) : all$_words_list = []forkeyindictl : all_words_list.append(key)forkeyindic$

$all_words_list.append(key)all_words_list_size = len(all_words_list)$

v1 = np.zeros(all$_words_list_size, dtype = np.int)v2 = np.zeros(all_words_list_size, dtype =$

$np.int)i = 0for(key)inall_words_list : v1[i] = dictl.get(key, 0)v2[i] = dict2.get(key, 0)i =$

$i + 1returncos_sim(v1, v2)$

def stop(text):

from nltk.corpus import stopwords from nltk.tokenize import word$_tokenizeimportnumpyasnpimportn$

def process(file): raw = open(file).read() tokens = word$_tokenize(raw)words = [w.lower()forwintoke$

$nltk.PorterStemmer()stemmed_tokens = [porter.stem(t)fortinwords]Removingstopwordsstop_word$

$set(stopwords.words('english'))filtered_tokens = [wforwinstemmed_tokensifnotwinstop_words]coun$

$nltk.defaultdict(int)forwordinfiltered_tokens : count[word]+ = 1returncount;$

def cos$_sim(a, b) : dot_product = np.dot(a, b)norm_a = np.linalg.norm(a)norm_b =$

$np.linalg.norm(b)returndot_product/(norm_a * norm_b)$

def getsimilarity(dictl, dict2): all$_words_list = []forkeyindictl : all_words_list.append(key)forkeyindic$

$all_words_list.append(key)all_words_list_size = len(all_words_list)$

v1 = np.zeros(all$_words_list_size, dtype = np.int)v2 = np.zeros(all_words_list_size, dtype =$

$np.int)i = 0for(key)inall_words_list : v1[i] = dictl.get(key, 0)v2[i] = dict2.get(key, 0)i =$

$i + 1returncos_sim(v1, v2)$

example$_sent = text.lower()example_sent = str(example_sent).replace('-','')example_sent =$

$str(example_sent).replace('_{','')stop_words=set(stopwords.words('english'))word_tokens=word_tokenize(example_sent)}$

filtered$_sentence = [wforwinword_tokensifnotwinstop_words]$

filtered$_sentence = []$

for w in word$_tokens : ifwnotinstop_words : filtered_sentence.append(w)$

return filtered$_sentencedefprocess(file) : tokens = wraw = fileord_tokenize(raw)words =$ $[w.lower()forwintokens]porter = nltk.PorterStemmer()stemmed_tokens = [porter.stem(t)fortinw$ $set(stopwords.words('english'))filtered_tokens = [wforwinstemmed_tokensifnotwinstop_words]coun$ $nltk.defaultdict(int)forwordinfiltered_tokens : count[word]+ = 1returncount; defcos_sim(a, b) :$ $dot_product = np.dot(a, b)norm_a = np.linalg.norm(a)norm_b = np.linalg.norm(b)returndot_product/($ $norm_b)$

@app.route('/send$_complaint', methods = ['post'])defsend_complaint() : uid = request.form['uid']c$ $request.form['com']$

qry = ”INSERT INTO ‘complaint‘ VALUES(NULL,val=(str(uid),com) iud(qry,val) return jsonify(’task’:”success”)

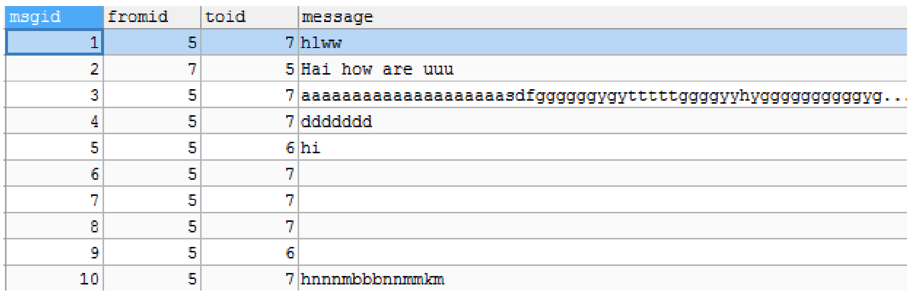
@app.route('/view$_complaint_reply', methods = ['post'])defview_complaint_reply() : lid =$ $request.form['uid']$

qry=”SELECT * FROM ‘complaint‘ WHERE ‘userid‘=value=(lid) res = androidselectall(qry,value) return jsonify(res)

if $_name_{=="}{}_{main_{"}:app.run(host="0.0.0.0",port=5000)}$

5.3 Database Design

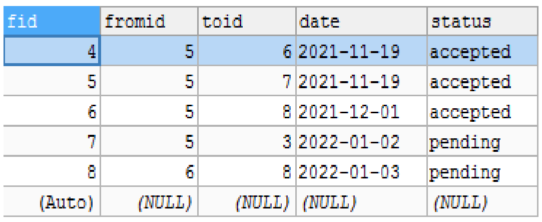
Chat



msgid	fromid	toid	message
1	5	7	hlww
2	7	5	Hai how are uuu
3	5	7	aaaaaaaaaaaaaaaaaaaasdfggggggygytttttggggyyhygggggggggyg...
4	5	7	ddddddd
5	5	6	hi
6	5	7	
7	5	7	
8	5	7	
9	5	6	
10	5	7	hnnnmbbbnnmmkm

Figure A.1: Table-1

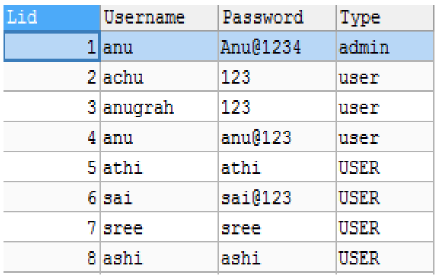
Friend Request



fid	fromid	toid	date	status
4	5	6	2021-11-19	accepted
5	5	7	2021-11-19	accepted
6	5	8	2021-12-01	accepted
7	5	3	2022-01-02	pending
8	6	8	2022-01-03	pending
(Auto)	*(NULL)*	*(NULL)*	*(NULL)*	*(NULL)*

Figure A.2: Table-2

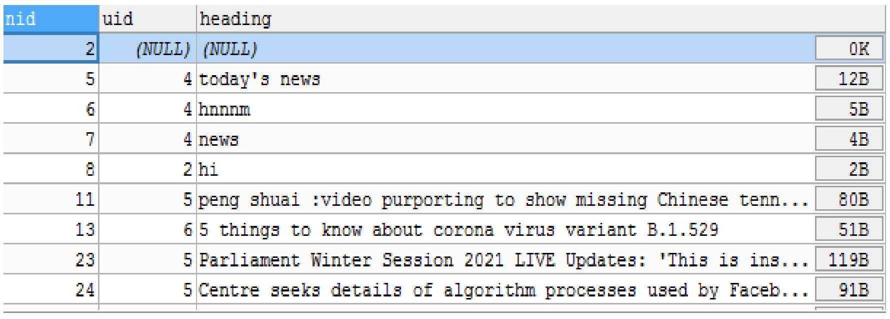
Login



Lid	Username	Password	Type
1	anu	Anu@1234	admin
2	achu	123	user
3	anugrah	123	user
4	anu	anu@123	user
5	athi	athi	USER
6	sai	sai@123	USER
7	sree	sree	USER
8	ashi	ashi	USER

Figure A.3: Table-3

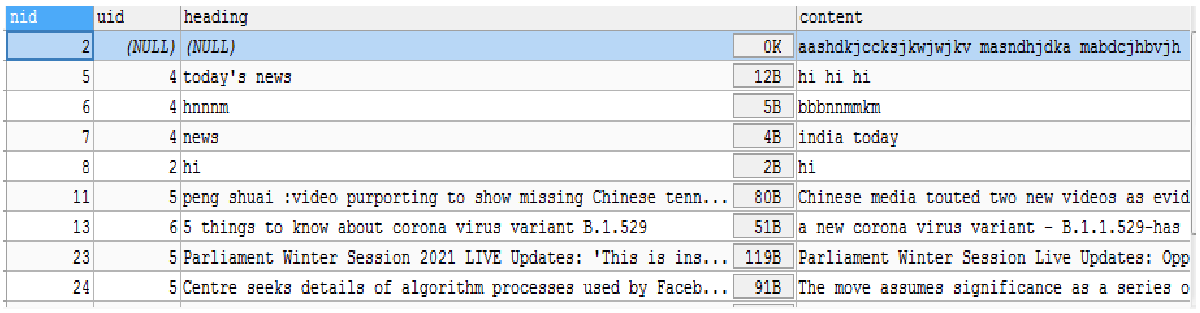
Feedback



nid	uid	heading	
2	(NULL)	(NULL)	0K
5	4	today's news	12B
6	4	hnnnm	5B
7	4	news	4B
8	2	hi	2B
11	5	peng shuai :video purporting to show missing Chinese tenn...	80B
13	6	5 things to know about corona virus variant B.1.529	51B
23	5	Parliament Winter Session 2021 LIVE Updates: 'This is ins...	119B
24	5	Centre seeks details of algorithm processes used by Faceb...	91B

Figure A.4: Table-4

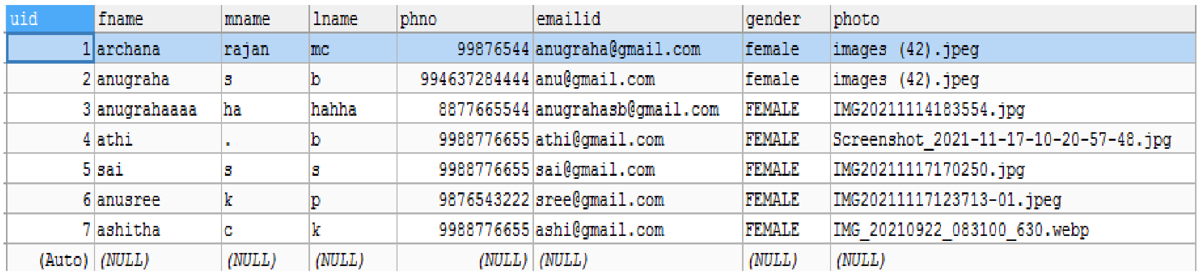
Post news



nid	uid	heading		content
2	(NULL)	(NULL)	0K	aashdkjccksjkwjwjkv masndhjdka mabdcjhbvjh
5	4	today's news	12B	hi hi hi
6	4	hnnnm	5B	bbbnnmmkm
7	4	news	4B	india today
8	2	hi	2B	hi
11	5	peng shuai :video purporting to show missing Chinese tenn...	80B	Chinese media touted two new videos as evid
13	6	5 things to know about corona virus variant B.1.529	51B	a new corona virus variant - B.1.1.529-has
23	5	Parliament Winter Session 2021 LIVE Updates: 'This is ins...	119B	Parliament Winter Session Live Updates: Opp
24	5	Centre seeks details of algorithm processes used by Faceb...	91B	The move assumes significance as a series o

Figure A.5: Table-5

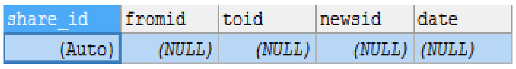
user registration



uid	fname	mname	lname	phno	emailid	gender	photo
1	archana	rajan	mc	99876544	anugraha@gmail.com	female	images (42).jpeg
2	anugraha	s	b	994637284444	anu@gmail.com	female	images (42).jpeg
3	anugrahaaaa	ha	hahha	8877665544	anugrahasb@gmail.com	FEMALE	IMG20211114183554.jpg
4	athi	.	b	9988776655	athi@gmail.com	FEMALE	Screenshot_2021-11-17-10-20-57-48.jpg
5	sai	s	s	9988776655	sai@gmail.com	FEMALE	IMG20211117170250.jpg
6	anusree	k	p	9876543222	sree@gmail.com	FEMALE	IMG20211117123713-01.jpeg
7	ashitha	c	k	9988776655	ashi@gmail.com	FEMALE	IMG_20210922_083100_630.webp
(Auto)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

Figure A.6: Table-6

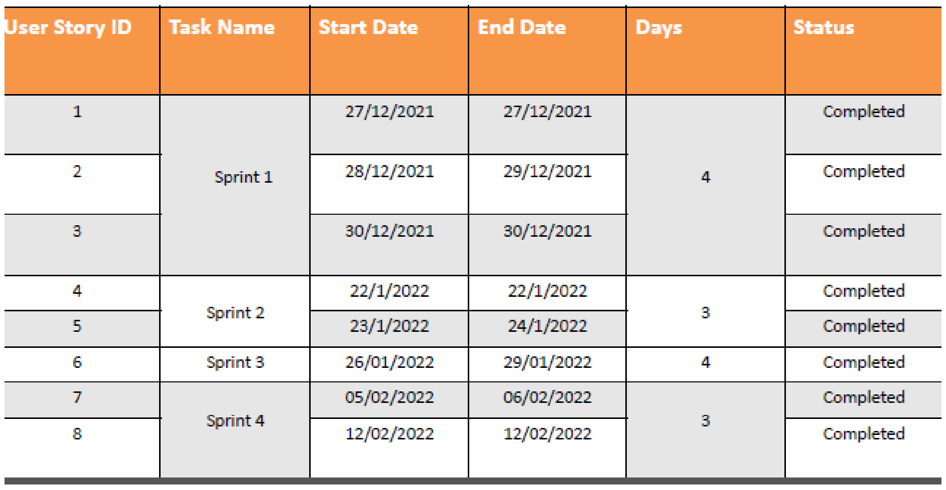
share news



share_id	fromid	toid	newsid	date
(Auto)	(NULL)	(NULL)	(NULL)	(NULL)

Figure A.7: Table-7

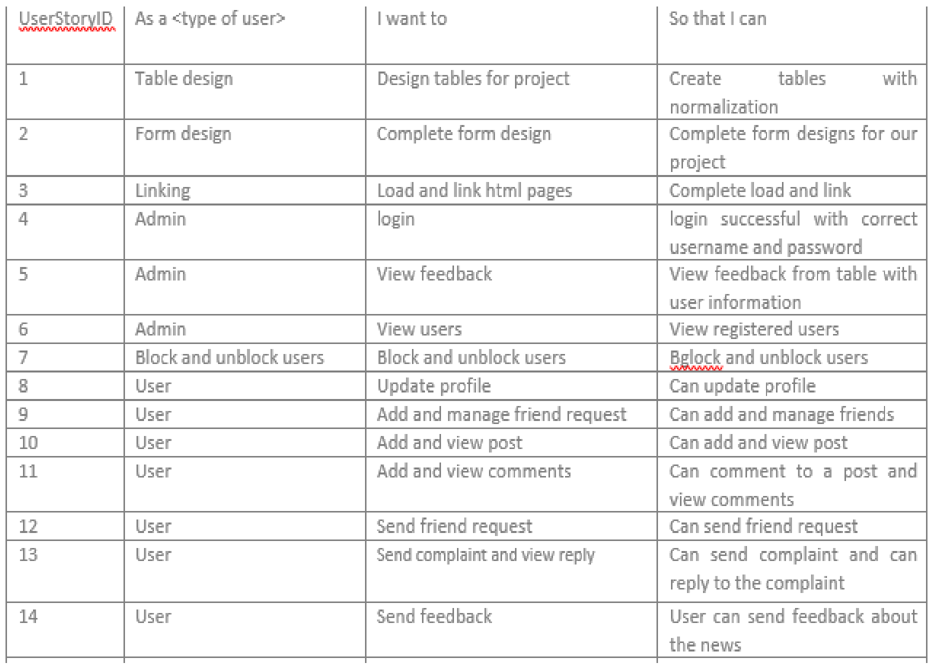
Project Plan



User Story ID	Task Name	Start Date	End Date	Days	Status
1	Sprint 1	27/12/2021	27/12/2021	4	Completed
2		28/12/2021	29/12/2021		Completed
3		30/12/2021	30/12/2021		Completed
4	Sprint 2	22/1/2022	22/1/2022	3	Completed
5		23/1/2022	24/1/2022		Completed
6	Sprint 3	26/01/2022	29/01/2022	4	Completed
7	Sprint 4	05/02/2022	06/02/2022	3	Completed
8		12/02/2022	12/02/2022		Completed

Figure A.8: Project Plan

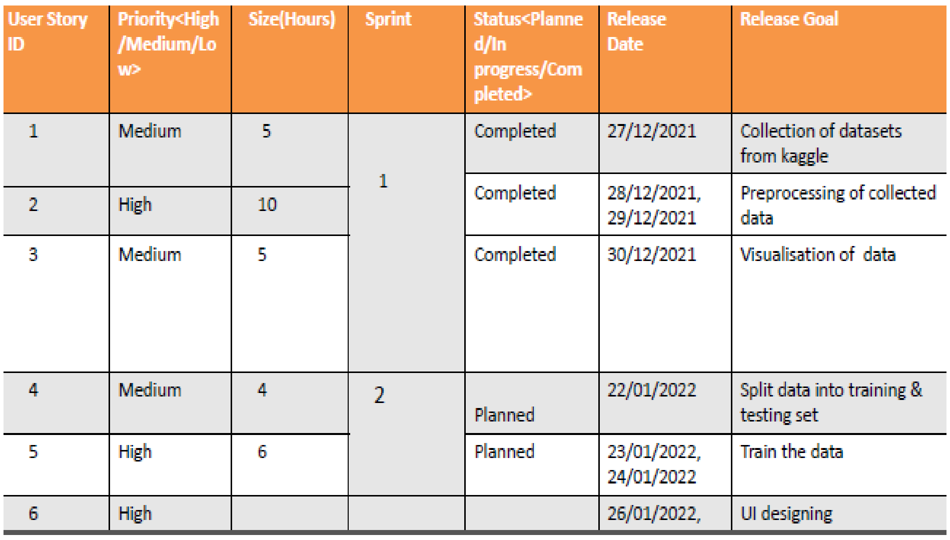
User Story



UserStoryID	As a <type of user>	I want to	So that I can
1	Table design	Design tables for project	Create tables with normalization
2	Form design	Complete form design	Complete form designs for our project
3	Linking	Load and link html pages	Complete load and link
4	Admin	login	login successful with correct username and password
5	Admin	View feedback	View feedback from table with user information
6	Admin	View users	View registered users
7	Block and unblock users	Block and unblock users	Bglock and unblock users
8	User	Update profile	Can update profile
9	User	Add and manage friend request	Can add and manage friends
10	User	Add and view post	Can add and view post
11	User	Add and view comments	Can comment to a post and view comments
12	User	Send friend request	Can send friend request
13	User	Send complaint and view reply	Can send complaint and can reply to the complaint
14	User	Send feedback	User can send feedback about the news

Figure A.9: User Story

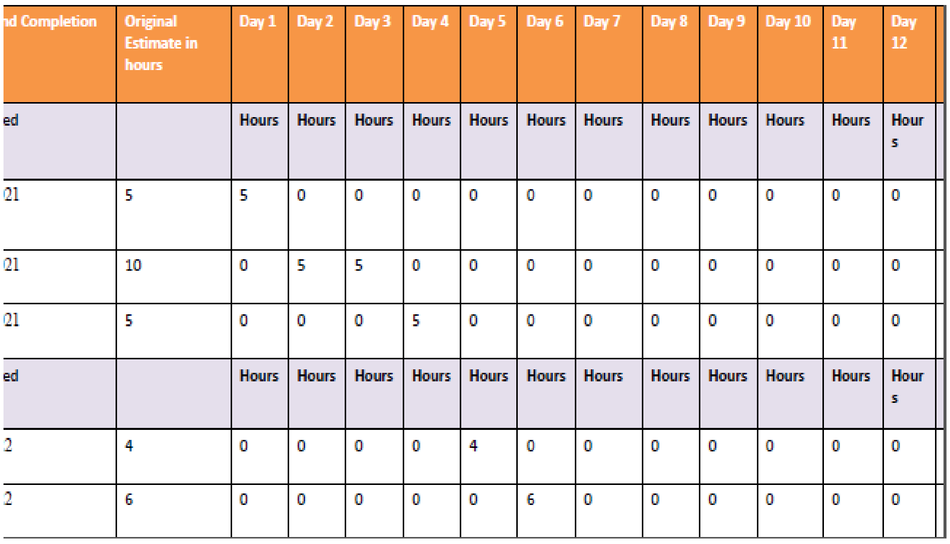
Product Backlog



User Story ID	Priority<High /Medium/Low>	Size(Hours)	Sprint	Status<Planned/In progress/Completed>	Release Date	Release Goal
1	Medium	5	1	Completed	27/12/2021	Collection of datasets from kaggle
2	High	10		Completed	28/12/2021, 29/12/2021	Preprocessing of collected data
3	Medium	5		Completed	30/12/2021	Visualisation of data
4	Medium	4	2	Planned	22/01/2022	Split data into training & testing set
5	High	6		Planned	23/01/2022, 24/01/2022	Train the data
6	High				26/01/2022,	UI designing

Figure A.10: Product Backlog

Sprint Plan



nd Completion	Original Estimate in hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12
ed		Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours
21	5	5	0	0	0	0	0	0	0	0	0	0	0
21	10	0	5	5	0	0	0	0	0	0	0	0	0
21	5	0	0	0	5	0	0	0	0	0	0	0	0
ed		Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours
2	4	0	0	0	0	4	0	0	0	0	0	0	0
2	6	0	0	0	0	0	6	0	0	0	0	0	0

Figure A.11: Sprint Plan

5.4 Admin

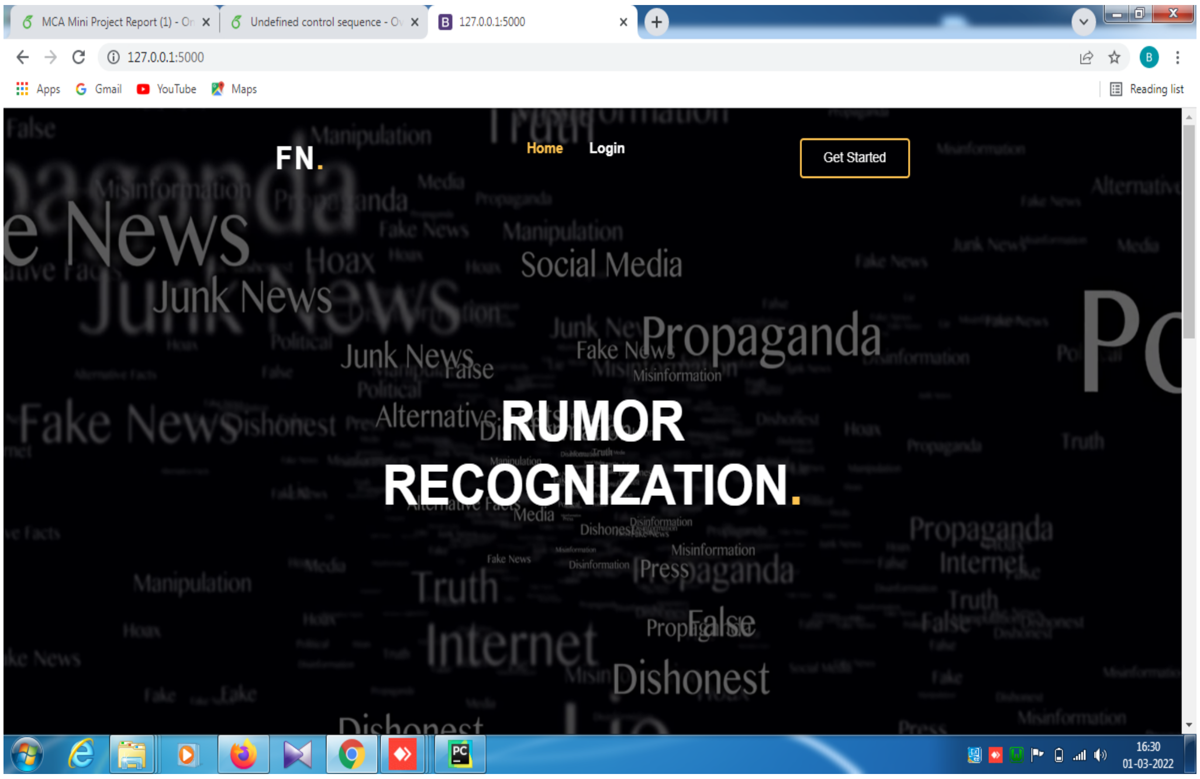


Figure A.12: home page

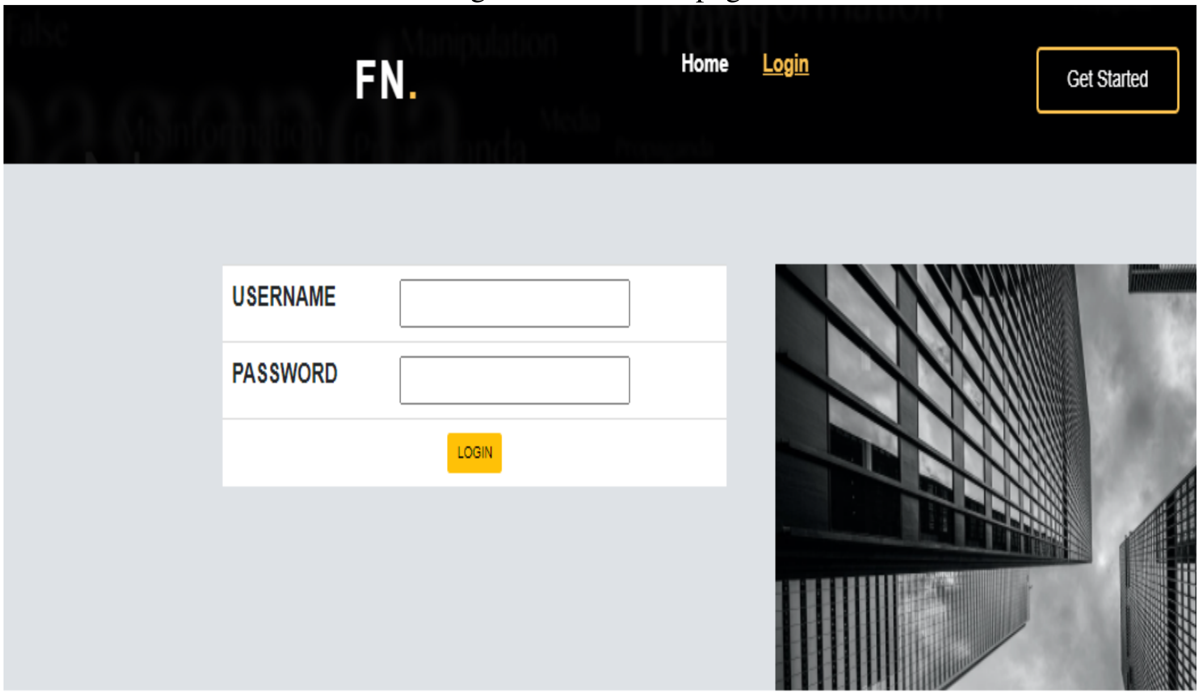


Figure A.13: login

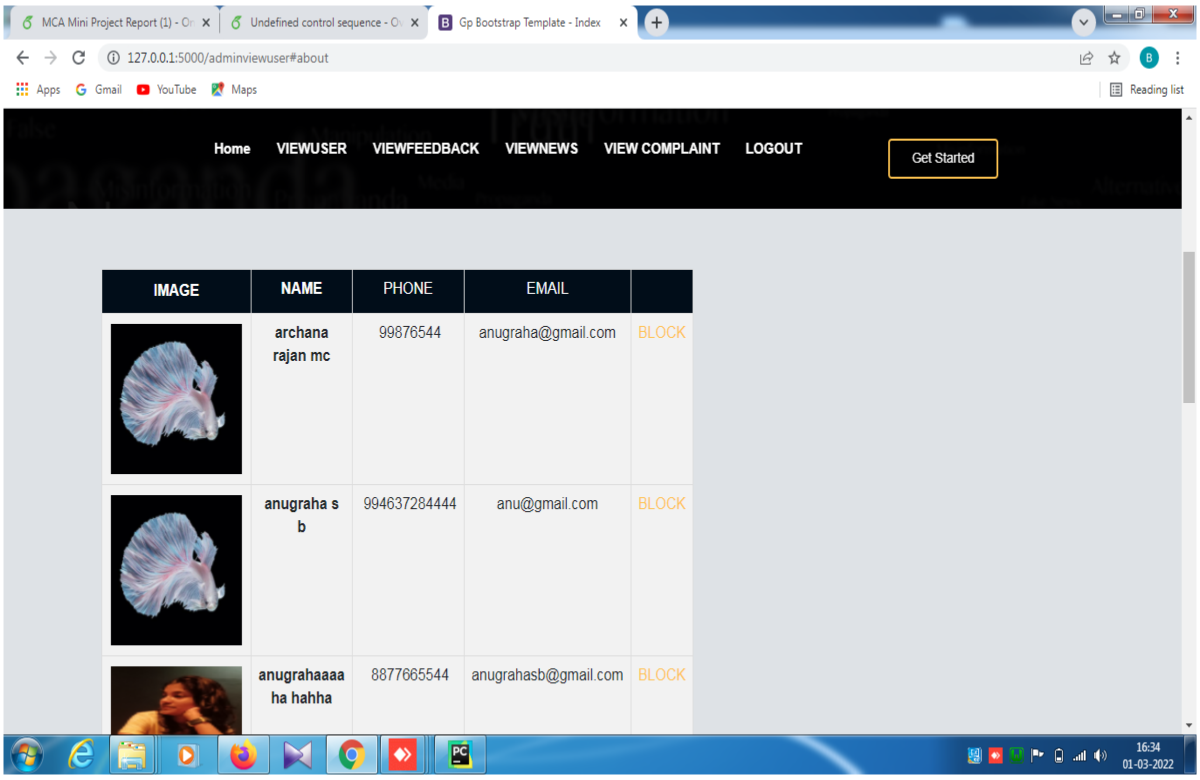
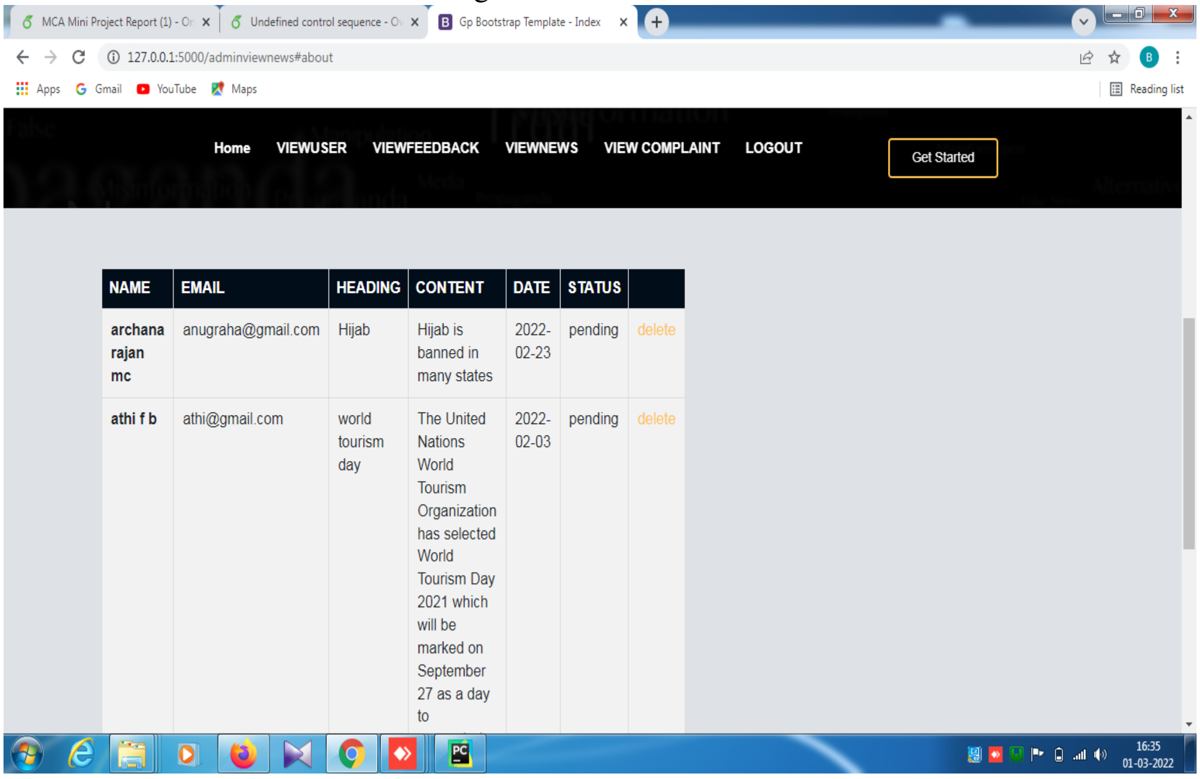

MCA Mini Project Report (1) - Or
Undefined control sequence - O
Gp Bootstrap Template - Index
127.0.0.1:5000/adminviewuser#about
Apps
Gmail
YouTube
Maps
Reading list
Home
VIEWUSER
VIEWFEEDBACK
VIEWNEWS
VIEW COMPLAINT
LOGOUT
Get Started
16:34
01-03-2022

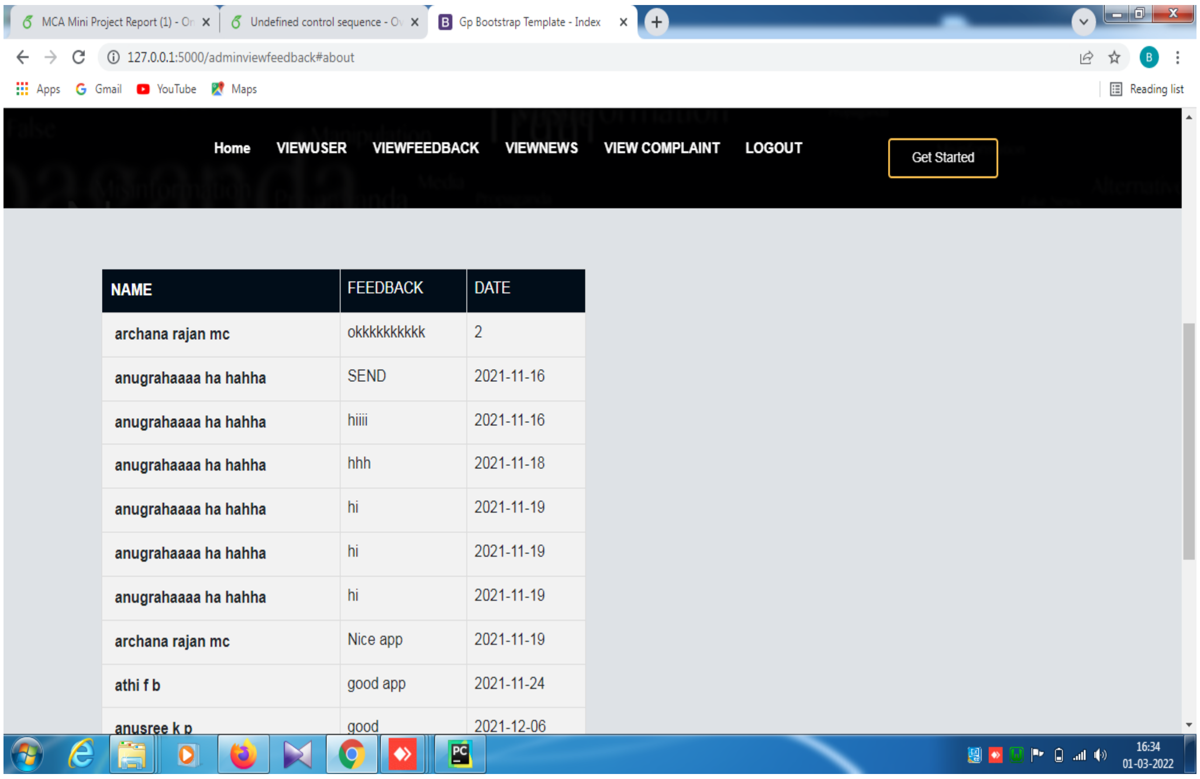
IMAGE	NAME	PHONE	EMAIL	
	archana rajan mc	99876544	anugraha@gmail.com	BLOCK
	anugraha s b	994637284444	anu@gmail.com	BLOCK
	anugrahaaaa ha hahha	8877665544	anugrahasb@gmail.com	BLOCK

Figure A.14: view user


MCA Mini Project Report (1) - Or
Undefined control sequence - O
Gp Bootstrap Template - Index
127.0.0.1:5000/adminviewnews#about
Apps
Gmail
YouTube
Maps
Reading list
Home
VIEWUSER
VIEWFEEDBACK
VIEWNEWS
VIEW COMPLAINT
LOGOUT
Get Started
16:35
01-03-2022

NAME	EMAIL	HEADING	CONTENT	DATE	STATUS	
archana rajan mc	anugraha@gmail.com	Hijab	Hijab is banned in many states	2022-02-23	pending	delete
athi f b	athi@gmail.com	world tourism day	The United Nations World Tourism Organization has selected World Tourism Day 2021 which will be marked on September 27 as a day to	2022-02-03	pending	delete

Figure A.15: viewnews


MCA Mini Project Report (1) - Or
Undefined control sequence - Ov
Gp Bootstrap Template - Index
127.0.0.1:5000/adminviewfeedback#about
Apps
Gmail
YouTube
Maps
Reading list
Home
VIEWUSER
VIEWFEEDBACK
VIEWNEWS
VIEW COMPLAINT
LOGOUT
Get Started
16:34
01-03-2022

NAME	FEEDBACK	DATE
archana rajan mc	okkkkkkkkkk	2
anugrahaaaa ha hahha	SEND	2021-11-16
anugrahaaaa ha hahha	hiiii	2021-11-16
anugrahaaaa ha hahha	hhh	2021-11-18
anugrahaaaa ha hahha	hi	2021-11-19
anugrahaaaa ha hahha	hi	2021-11-19
anugrahaaaa ha hahha	hi	2021-11-19
archana rajan mc	Nice app	2021-11-19
athi f b	good app	2021-11-24
anusree k p	good	2021-12-06

Figure A.16: viewfeedback

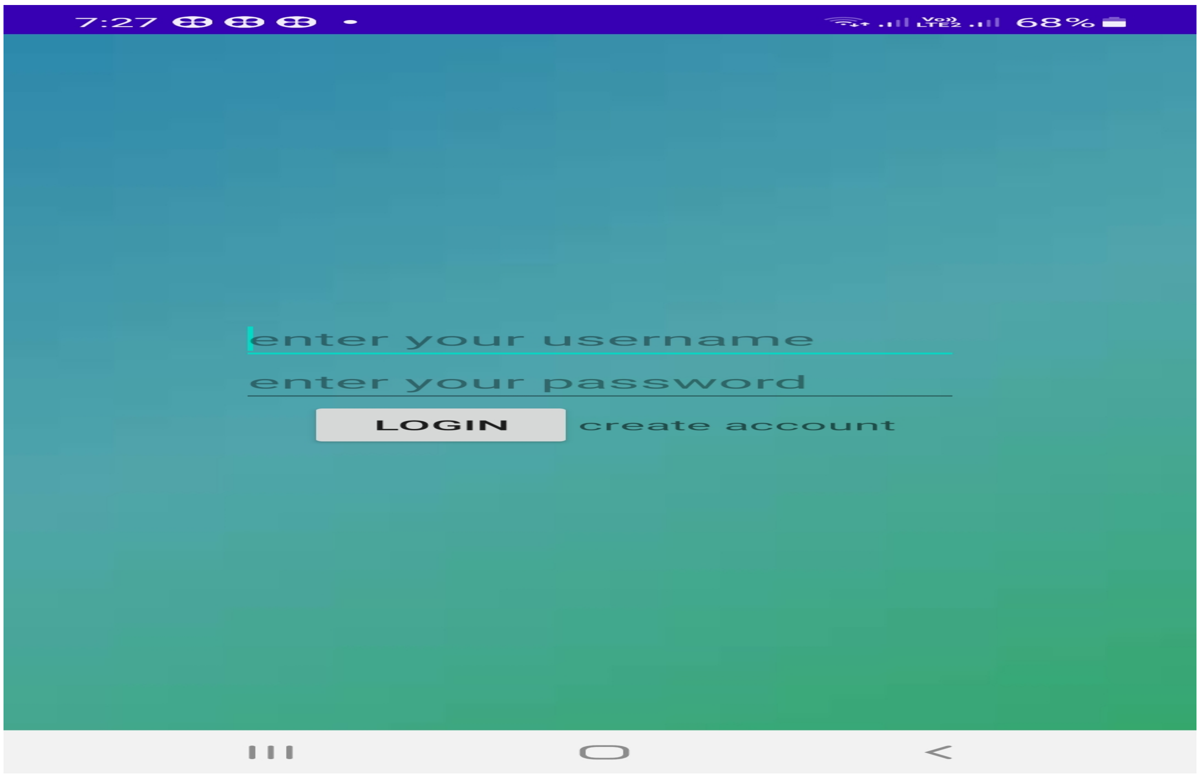


Figure A.17: userlogin

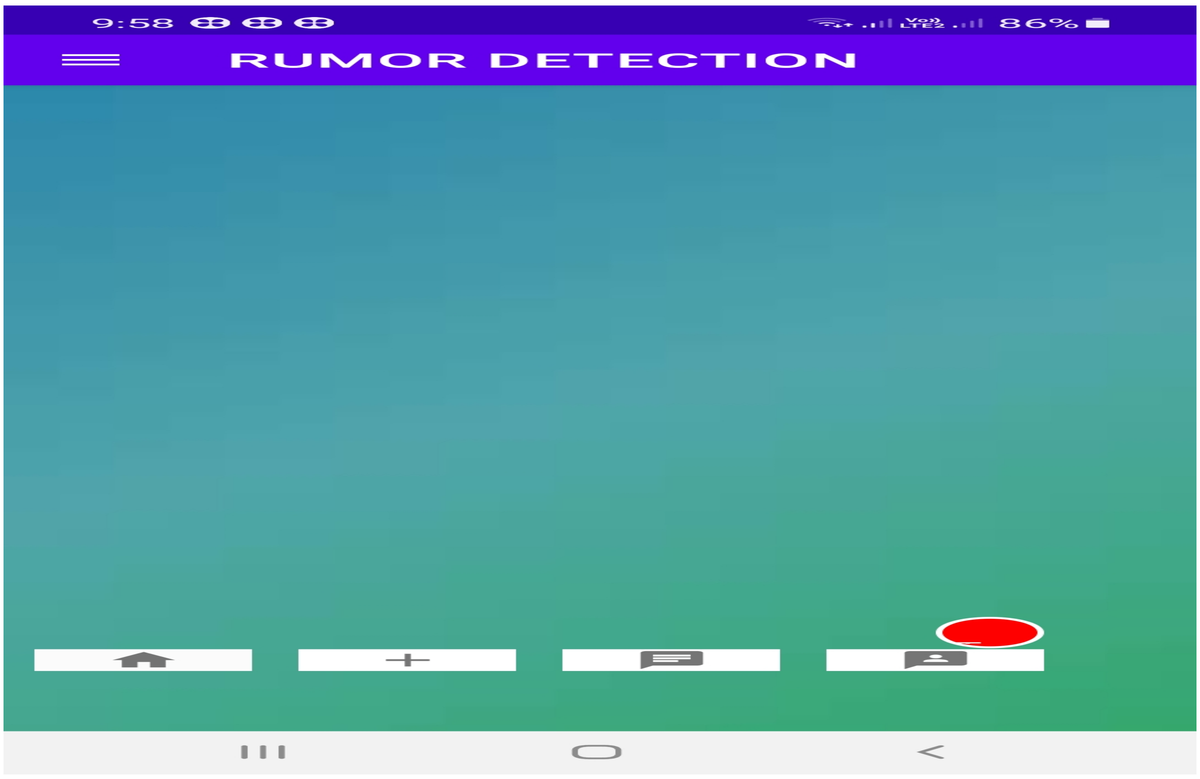


Figure A.18: home

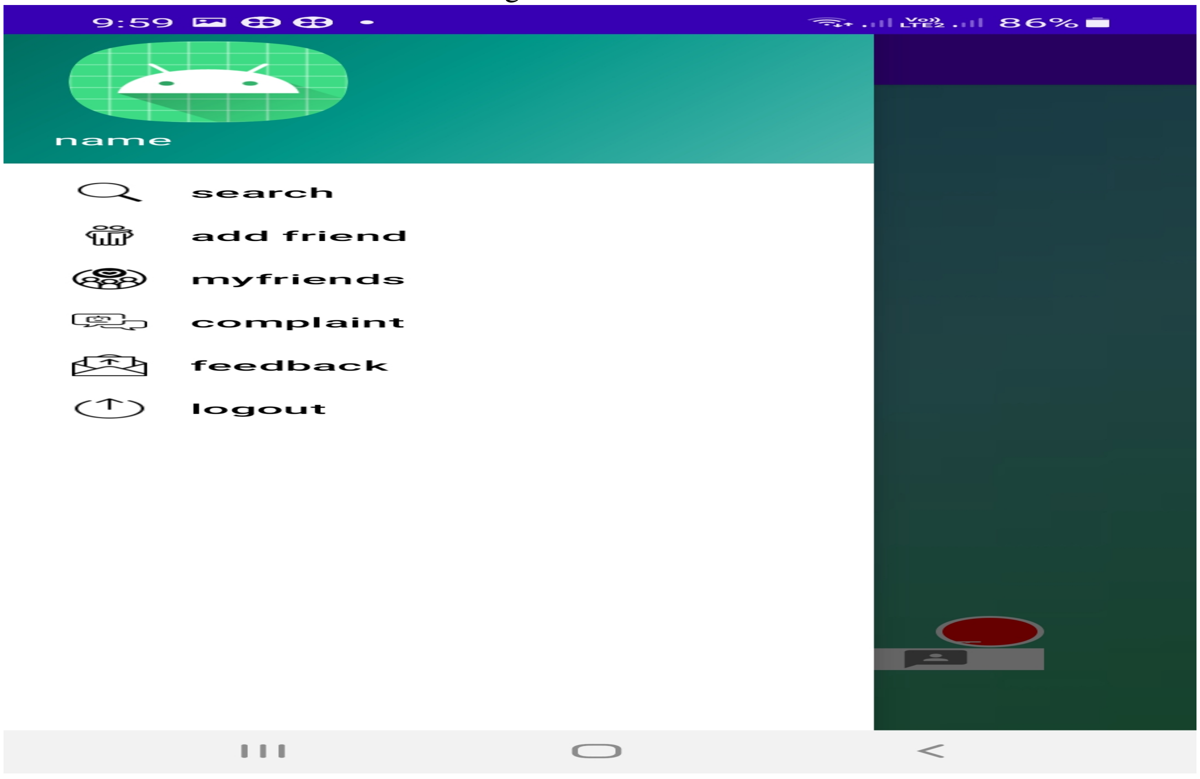


Figure A.19: searches

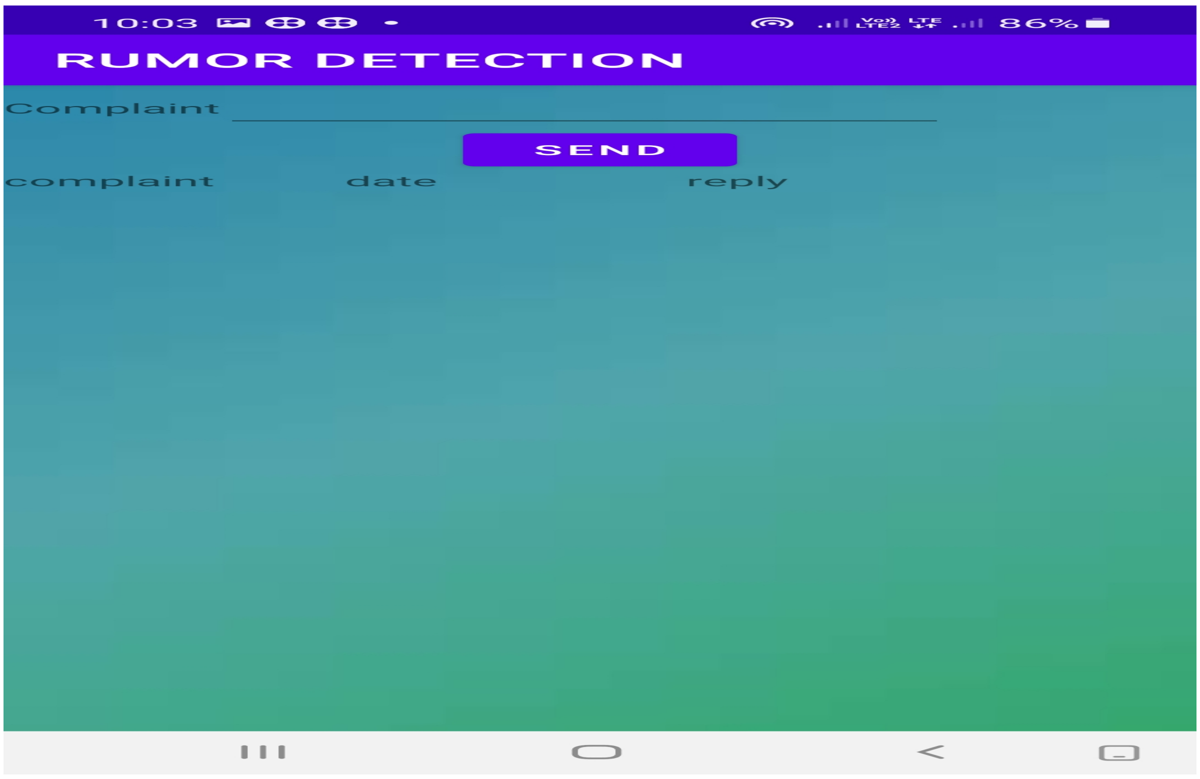
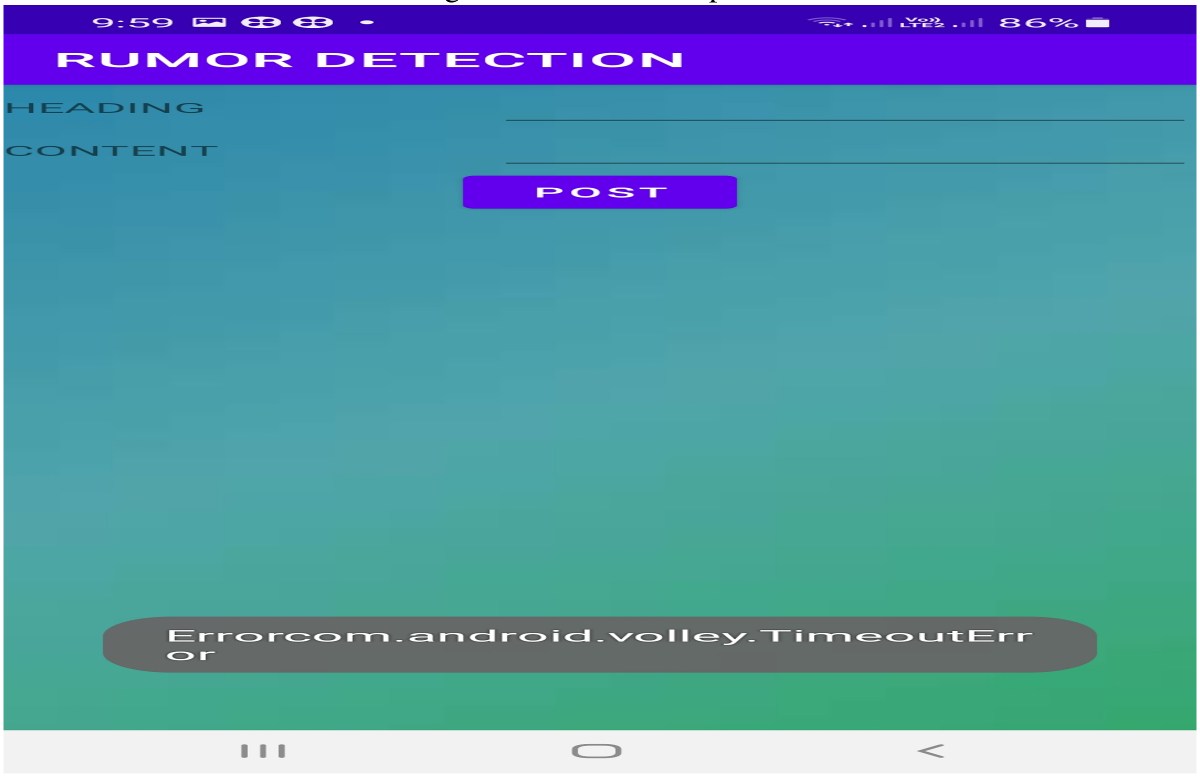


Figure A.20: usercomplaint



Errorcom.android.volley.TimeoutErr
or

Figure A.21: newspost

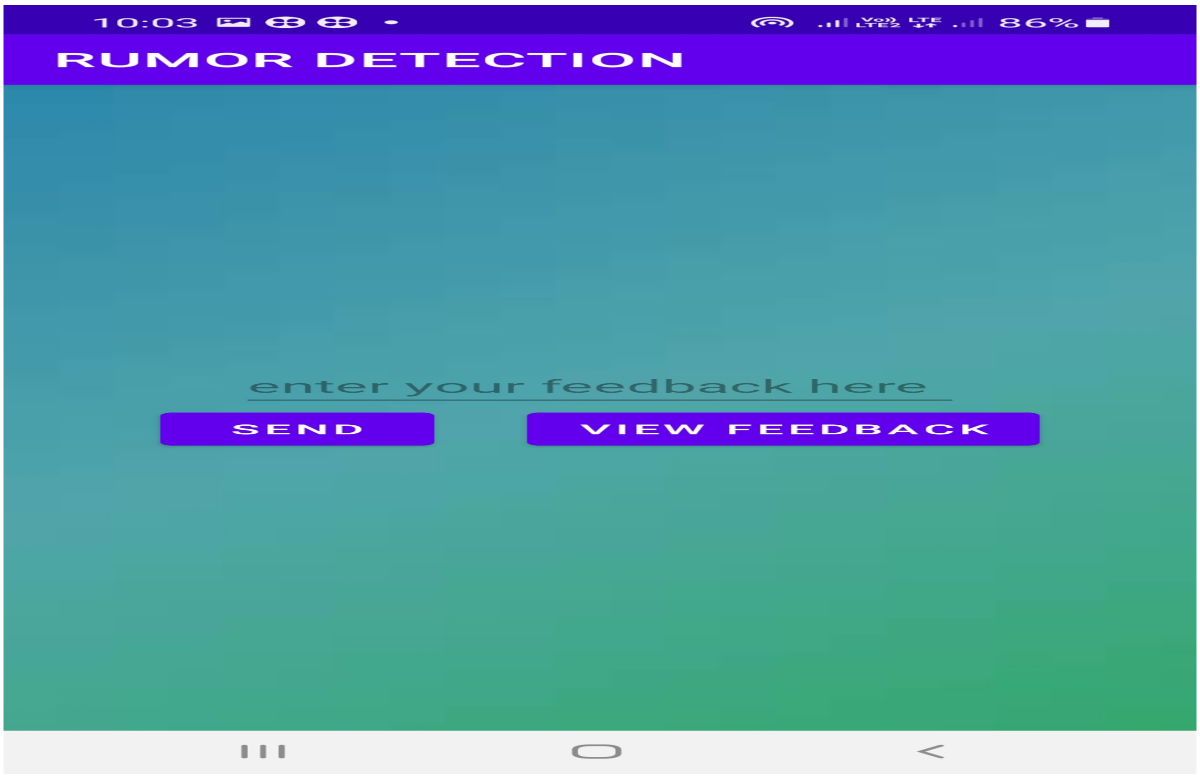


Figure A.22: feedback